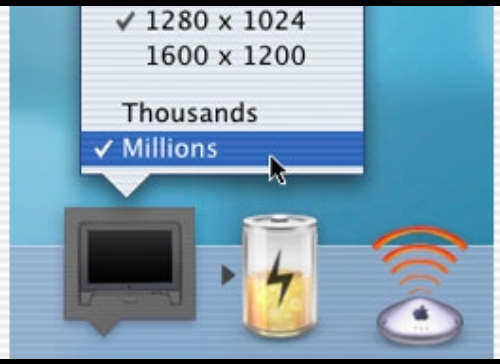
Session 204

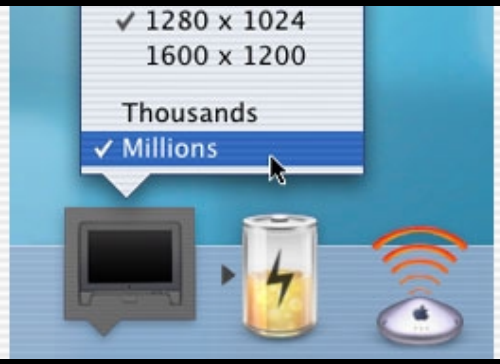
I/O Kit: Storage Drivers



Craig Keithley
USB & FireWire Technology Manager
Worldwide Developer Relations

Session 204

I/O Kit: Storage Drivers



Craig Marciniak
Senior Engineer
Mass Storage Software

Audience

- Hardware Vendors interested in bringing their hardware to Mac OS X
- Application developers who need to access hardware from user space

What You'll Learn

- How Mac OS X Mass Storage works
- How the driver stacks are instantiated
- How to subclass support for unique device features
- How to talk to devices from applications
- Where to get more information

Basic I/O Kit Concepts

- Drivers are Kernel Extensions
- They're written with a restricted subset of C++
- Families versus Drivers
- Drivers in the parlance of I/O Kit are "Leaf Classes" of the family drivers

Basic I/O Kit Concepts (Cont.)

- The Registry (IORegistryExplorer, ioreg)
- Matching
- Getting more information
  - Other sessions
  - URLs
  - Inside Mac OS X: Kernel Environment
  - Darwin

The Simplified Driver Stack

The Simplified Driver Stack

Physical Interconnect Layer

The Simplified Driver Stack

Transport Driver Layer

Physical Interconnect Layer

The Simplified Driver Stack

Device Services Layer

Transport Driver Layer

Physical Interconnect Layer

The Simplified Driver Stack

Clients

Device Services Layer

Transport Driver Layer

Physical Interconnect Layer

The Transport Driver Layer

Clients

Device Services Layer

Transport Driver Layer

Physical Interconnect Layer

What We Learned From Mac OS 9

- Drivers are monolithic—no reusability, variance in implementation, etc...
- Required detailed knowledge of the file system, partition information, etc...

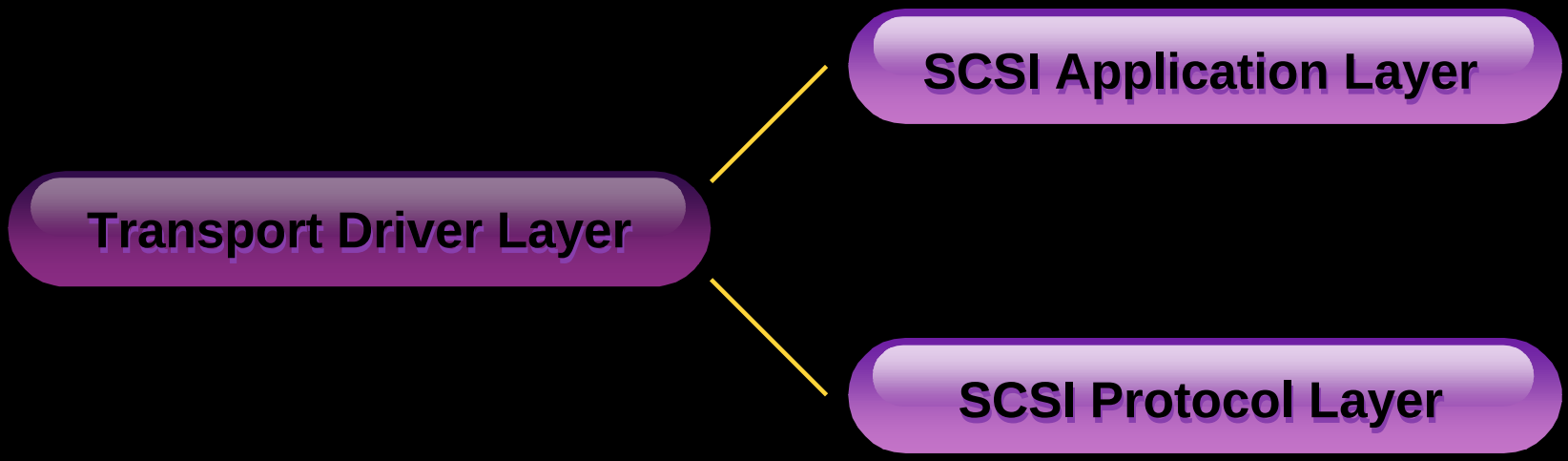
What We Did for Mac OS X

- We want to reuse commonality across busses
- I/O Kit provides an object-oriented framework to achieve this

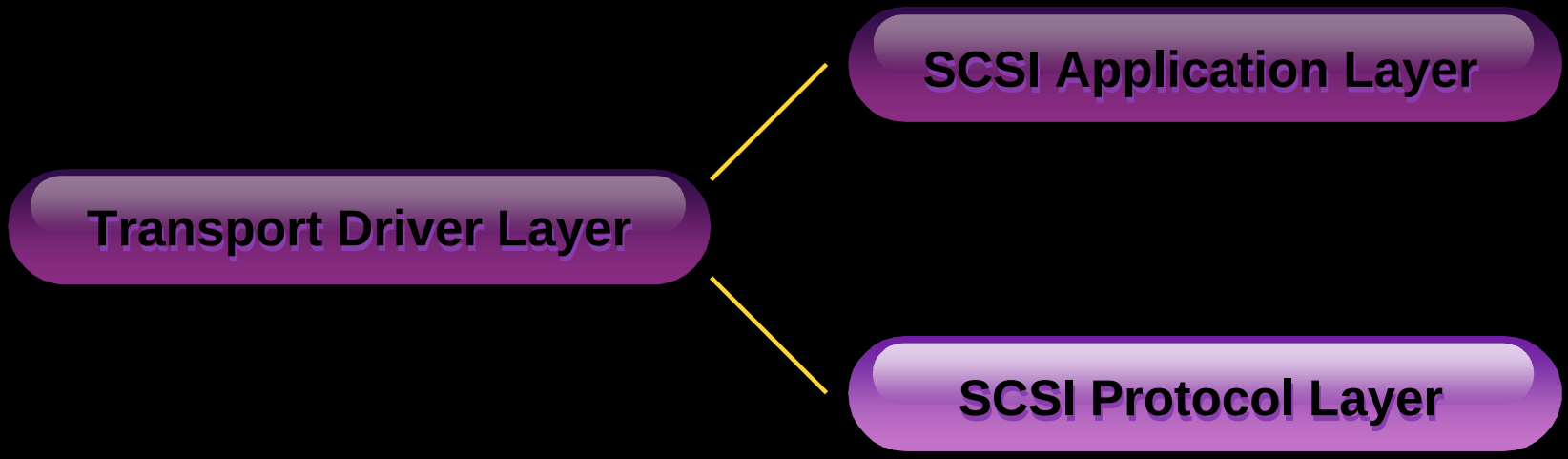
The SCSI Architecture Model

- Why we selected SAM
- SAM is a documented guideline for implementing a system across all SCSI interconnect and protocol environments
- SAM defines the functional partitioning of the SCSI command sets and SCSI protocol standards
- ftp://ftp.t10.org/t10/drafts/sam2/

The Transport Driver Layer



SCSI Protocol Layer



IOUSBMassStorageClass,
IOATAPIProtocolTransport,
IOFireWireSerialBusProtocolTransport, etc..

The SCSI Application Layer

SCSI Application Layer

SCSI Protocol Layer

Logical Unit Driver

IOSCSIPeripheralDeviceType00,
IOSCSIPeripheralDeviceType05,
IOSCSIPeripheralDeviceType07,
IOSCSIPeripheralDeviceType0E

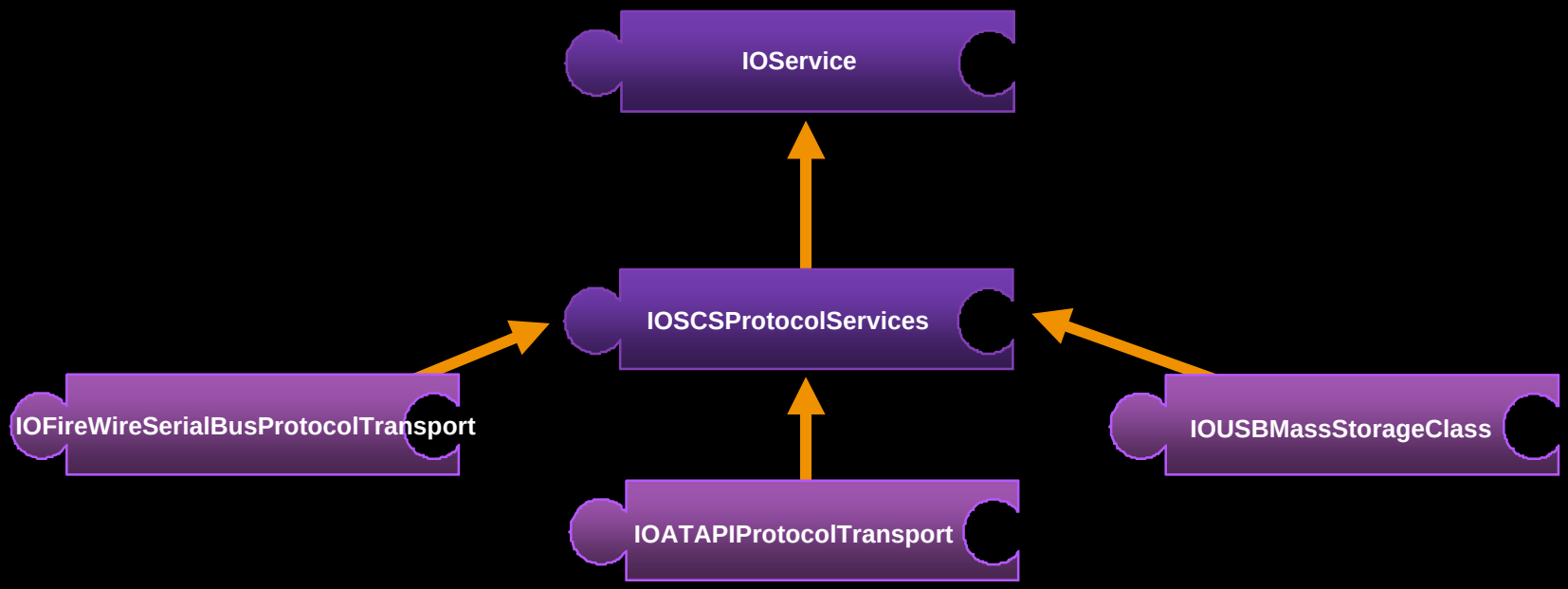
Peripheral Device Type Nub

IOSCSIPeripheralDeviceNub

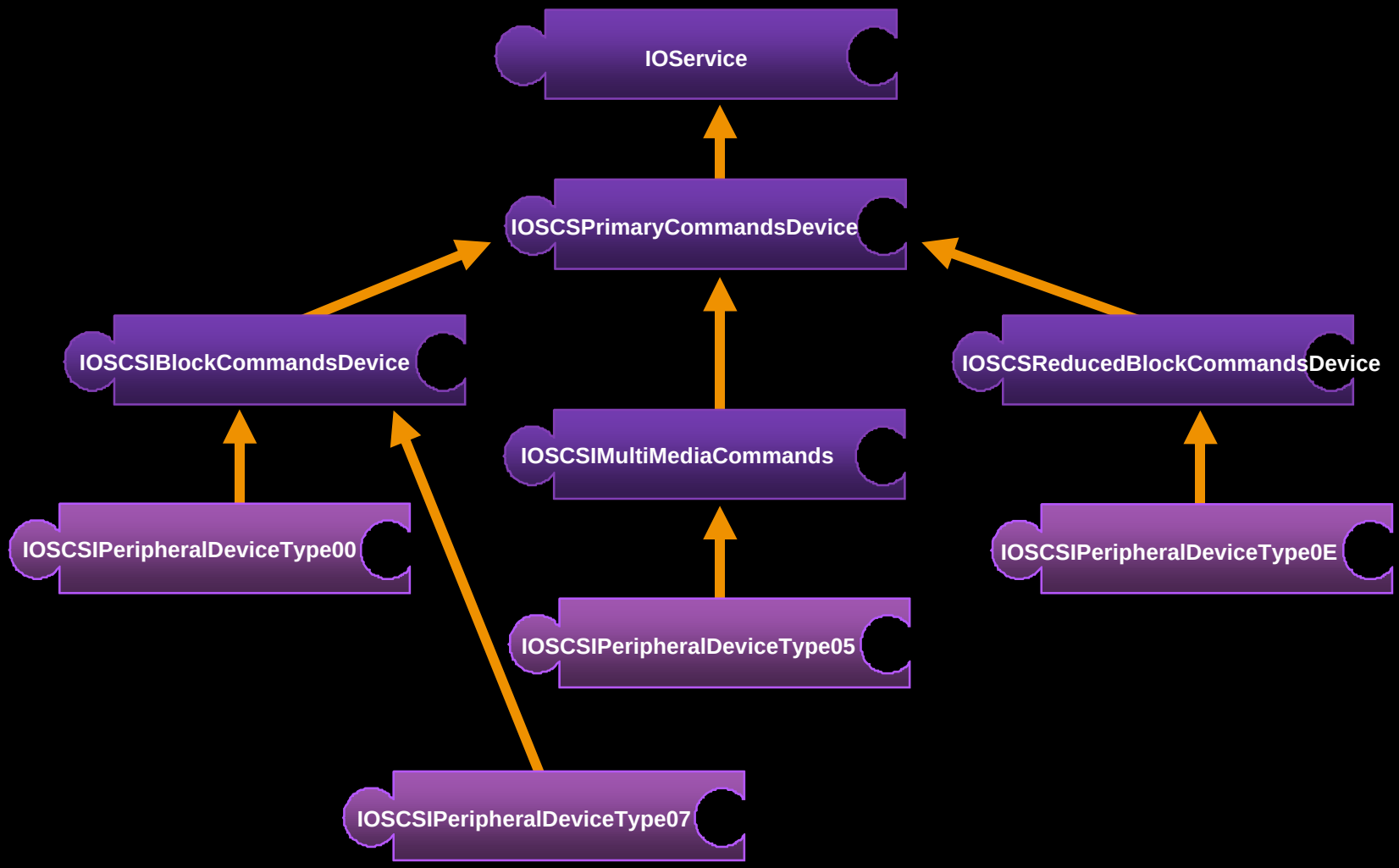
The SCSI Architecture Model Family

Device Services	IOBlockStorageServices, IOReducedBlockServices, IOCompactDiskServices, IODVDServices
Logical Unit Driver	IOSCSIPeripheralDeviceType00, IOSCSIPeripheralDeviceType05, IOSCSIPeripheralDeviceType07, IOSCSIPeripheralDeviceType0E
Peripheral Device Type Nub	IOSCSIPeripheralDeviceNub
SCSI Protocol Driver	IOUSBMassStorageClass, OATAPIProtocolTransport, IOFireWireSerialBusProtocolTransport

SCSI Protocol Layer Object Hierarchy



SCSI Application Layer Object Hierarchy



Example Stack

IOMedia

IOBlockStorageDriver

IOBlockStorageServices

IOSCSIPeripheralDeviceNub

IOFireWireSBP2

IOFireWireFamily

IOSCSIPeripheralDeviceType00

IOSCSIBlockCommandsDevice

IOSCSIPrimaryCommandsDevice

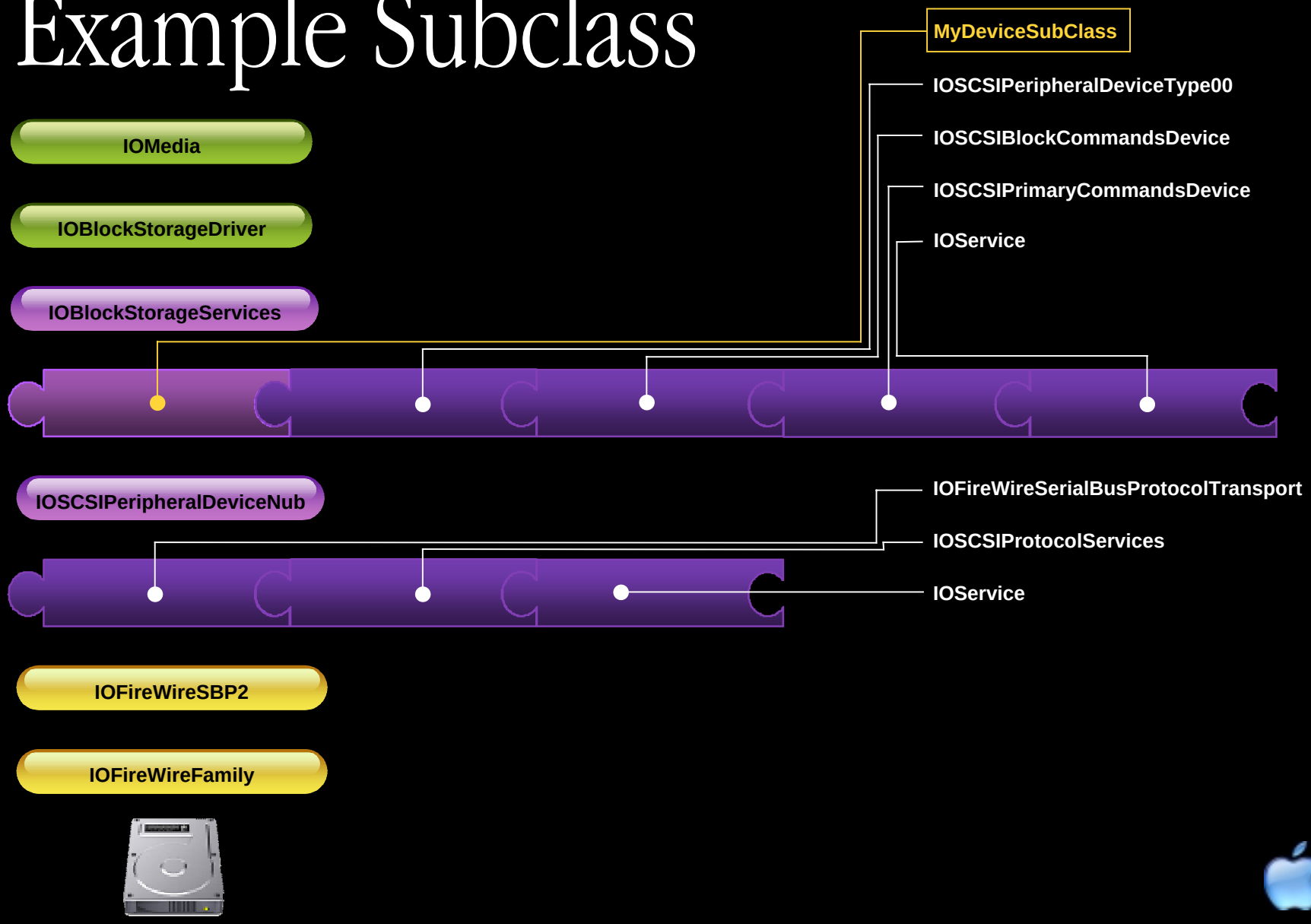
IOService

IOFireWireSerialBusProtocolTransport

IOSCSIProtocolServices

IOService

Example Subclass



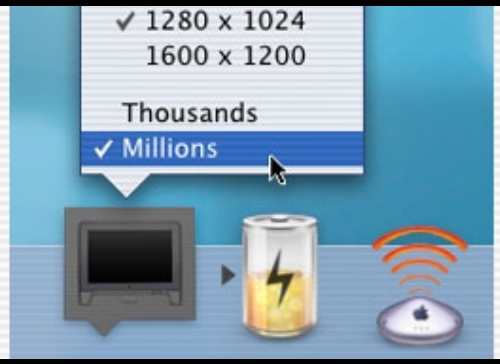
Matching

- The highest probe score wins
- The more properties that the logical unit driver's personality matches—the higher the probe score
  - Peripheral Device Type
  - Vendor Identification
  - Product Identification
  - Product Revision Level

Overriding the Probe Method

- Overriding the probe method
- Sometimes you need to parse bits in a property
- Be careful when referencing member variables
- See example in AppleFireWireMassStorageDrivers

Demo



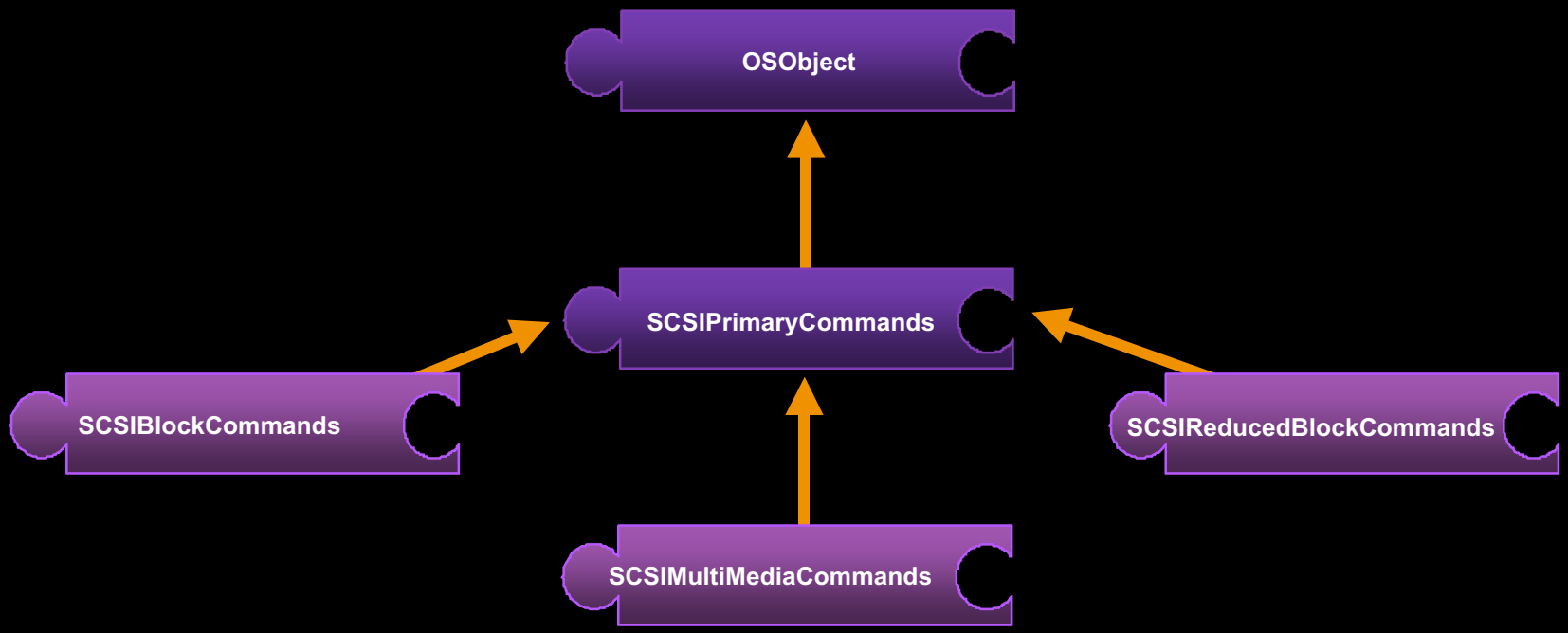
Chris Sarcone
Software Engineer
Mass Storage Software

The SCSITask Object

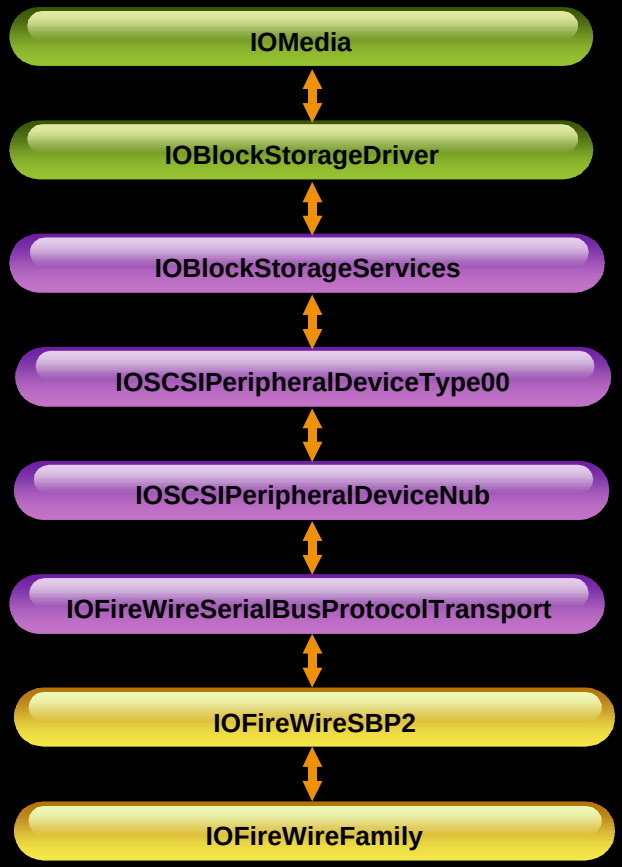
- An object that encapsulates everything needed to execute and complete a command request
  - Command Descriptor Block
  - Data Buffers
  - Command Byte Count
  - Sense Data
  - Etc…
- See SAM section 5, SCSI Command Model

The Command Set Builders

Utility classes for building command set specific commands in the SCSI Application Layer



How SCSITasks are Processed



I/O Request Generated

Appropriate SCSITask is created

SCSITask is packaged for target bus

Bus-specific command is processed

Device Compliance

- Don't worry about the hard stuff
  - Driver Life-cycle (hot-plugging)
  - Power Management
  - Etc…
- What we mean by device compliance
- Of course you can always override the entire driver

What's Supported?

- SAM support for:
  - ATAPI
  - FireWire
  - USB
- What about SCSIParallel?
- What about ATA?

Accessing Devices from Applications

- Whenever possible be in user space
  - Resources
  - Better Tools
  - System Stability
- Examples—Utilities that set device features, out of kernel drivers (Tape, Medium Changers, etc…)
- Negotiating the user-kernel boundary requires a user client

The SCSITask User Client

- Two access points
  - The peripheral device nub (IOSCSIPeripheral DeviceNub)—for any “other” peripheral device (e.g. Sequential Access Device Type 01)
  - Generic Services Layer for MultiMedia Commands devices (IOCompactDiscServices, and IODVDServices)

The SCSITask User Client

- Interfaces are through a COM-like CFPlugIn architecture
  - MMC Device Interface (Query for specific device information, and transition from non-exclusive to exclusive mode for MMC devices)
  - SCSITask Device Interface (create and release SCSITasks, add async callback handlers to CFRunLoop, etc…)
  - SCSITask Interface (build raw SCSITasks)

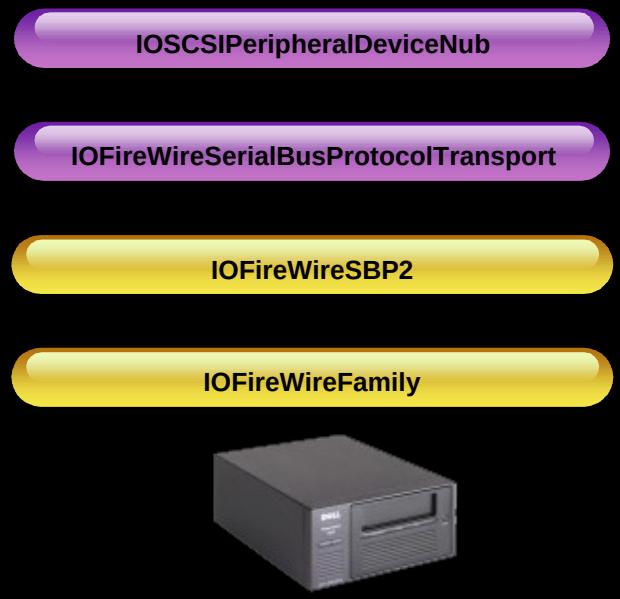
Exclusive Access Model

- Once instantiated the user client becomes the Logical Unit Driver
- There are no restrictions–the client has absolute control of the device
- Generic Services for MultiMedia
  - Unmount any mounted partitions
  - Obtain a reservation for the media

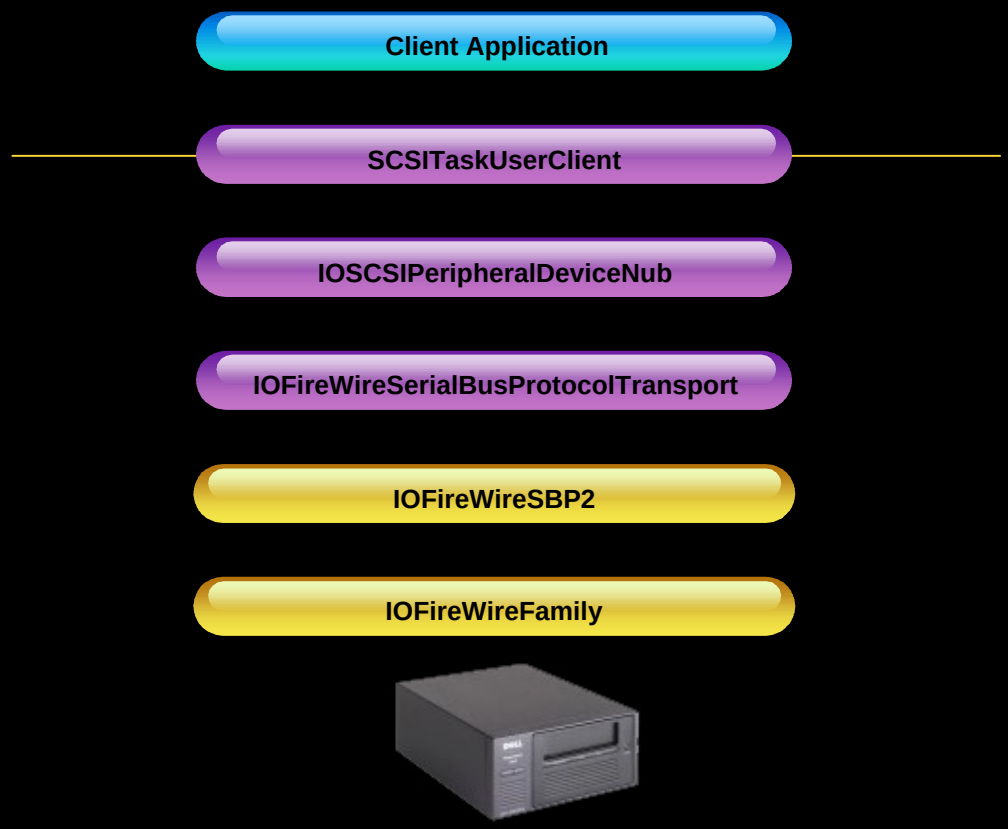
Non-Exclusive Access Model

- Generic Services for MultiMedia access point only
- Allows non-exclusive access to query the device for information
- API driven (i.e. no raw SCSITasks)
- If an existing user client has exclusive access non-exclusive commands are rejected

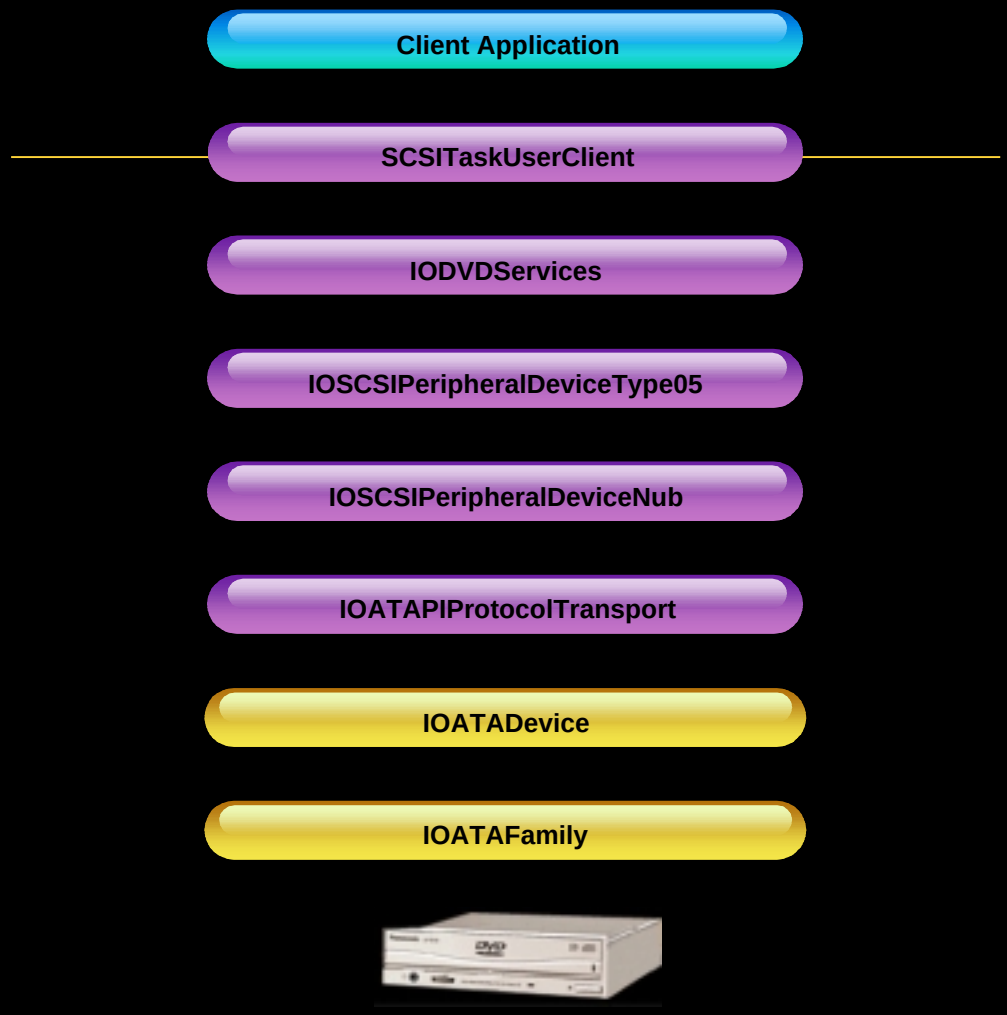
SCSITaskUserClient Example 1



SCSITaskUserClient Example 1



SCSITaskUserClient Example 2



Summary

- It really is easy to build off the existing driver architecture and bring your device on board in a short time

Resources

SCSI Architecture Model Specification
ftp://ftp.t10.org/t10/drafts/sam2/

Mac OS X Developer Information
http://developer.apple.com/macosx/

Mac OS X Accessing Hardware from Applications
http://developer.apple.com/techpubs/macosx/Kernel/kernel.html

Macintosh USB Development Information
http://developer.apple.com/hardware/usb/

Macintosh FireWire Development Information
http://developer.apple.com/hardware/firewire/

Related Sessions

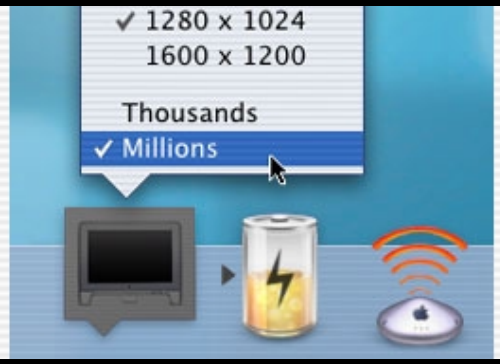
104 I/O Kit Update: An overview of I/O Kit in Mac OS X	Room A1 **Mon., 3:30pm**
205 I/O Kit: PCI Drivers & Open Firmware	Room C **Wed., 9:00am**
206 FireWire In-Depth: Programming details and Mac OS X	Civic **Wed., 2:00pm**
207 USB In-Depth: Details on creating USB drivers in Mac OS X	Civic **Wed., 3:30pm**

Who to Contact

Mac OS X Mass Storage Developer Mailing List
x_mass_storage@sam.apple.com

Craig Keithley
USB & FireWire Technology Manager
Worldwide Developer Relations
keithley@apple.com

Q&A



Craig Keithley
USB & FireWire Technology Manager
keithley@apple.com

http://developer.apple.com/events/urls.html

WWDC

Worldwide Developers Conference 2001